JOURNAL OF COMPUTATIONAL PHYSICS 77, 270–275 (1988)

Note

New Software for Large Dense Symmetric Generalized Eigenvalue Problems Using Secondary Storage*

1. INTRODUCTION

Quantum mechanical bandstructure computations require repeated computation of a large number of eigenvalues of a symmetric generalized eigenproblem. In previous work of the authors [7] the application of a block Lanczos algorithm, and a modification of the packed EISPACK [5, 12] routines, have been considered for the numerical solution of such eigenvalue problems arising in the theoretical calculation of pressure and volume at metallization of *BaTe* (see [14]). Both approaches were limited in the size of problem, which could be handled by the existing software. The largest feasible problem for each method on a 4 Mword Cray X-MP/24 was about of the order of 1900. Here we report numerical results with some new software, which has been developed recently. This software permits the user to fully utilize a fast secondary storage device such as the SSD. On a 128 Mword SSD now the solution of generalized eigenvalue problems of order up to 8000 is feasible. In this note we give a brief summary of the new software and present some performance results and a comparison with the results in [7].

We consider algorithms for the efficient solution of the symmetric generalized eigenproblem

$$AX = BX\Lambda. \tag{1}$$

The new software provides a true out-of-core implementation of some standard routines from EISPACK. The approach is to first reduce (1) to standard form by applying the Cholesky factor L of B as

$$(L^{-1}AL^{-T})(L^{T}X) = (L^{T}X)\Lambda$$

or

$$CY = Y\Lambda, \tag{2}$$

where

$$\begin{aligned} C &= L^{-1}AL^{-T} \\ Y &= L^{T}X. \end{aligned} \tag{3}$$

* This work is supported through NSF Grant ASC-8519354.

0021-9991/88 $3.00

The matrix B is factored using the block Cholesky algorithm where B is initially stored in secondary storage and is overwritten with its factor L [6, 13].

The next step in the standard EISPACK approach would be to reduce C to tridiagonal form using a sequence of Householder transformations. The application of Householder transformations to reduce C to tridiagonal form requires access to all of the unreduced portion of C for the reduction of each column. This access requirement would incur $O(n^3)$ I/O transfers. This amount of I/O is too high for an out-of-core implementation, since it would effectively limit the efficiency of the algorithm by the transfer speed to and from secondary storage.

Instead a block Householder transformation is used to reduce C to a band matrix, where the band matrix will have small enough bandwidth to be held in central memory. This reduction is accomplished by first symmetrically partitioning the matrix C into equal sized blocks with maximum block size p. Then for each block column we compute the QR factorization of the matrix comprised of the blocks which are below the diagonal block. The product of Householder transformations used to compute the QR factorization is accumulated in the WY representation described in [1]. This representation is then applied simultaneously to the left and the right of the remainder of the matrix. When all block columns except the last is reduced in this fashion, the result is a band matrix with bandwith p which is similar to the matrix C in (2). The I/O requirements of the block Householder transformation are of $O(n^3/p)$. Hence for any reasonable block size, the performance of the algorithm will no longer be I/O bound.

The banded eigenproblem is then transformed to tridiagonal form using an enhanced version of EISPACK subroutine BANDR. Finally EISPACK subroutine TQLRAT is used to compute the eigenvalues. Details of the modifications to BANDR, and a comparison of the performance of TQLRAT with a new algorithm by Dongarra and Sorensen [3] are given in a forthcoming paper [8].

Specified eigenvectors are computed by first computing the associated eigenvectors of the band matrix and then back transforming them to Y using the accumulated block Householder transformations. Application of L to Y back transforms Y to X, the eigenvectors of the original problem (1). The eigenvectors of the band matrix are computed using inverse iteration implemented in a much modified version of EISPACK subroutine BANDV.

This algorithm has been implemented in a pair of subroutines HSSXGV and HSSXG1 which, respectively, compute the eigenvalues and a specified set of eigenvectors of (1). The usage of these two subroutines as well as subroutines HSSXEV and HSSXE1, which compute the eigenvalues and specified eigenvectors of (2), is described in [9].

2. Resource Requirements

The new software makes extensive use of both the central processing unit and secondary storage of a computer. This section will discuss the requirements of the

algorithm for secondary storage, amount of I/O transfer between central memory and secondary storage, storage requirements for central memory, and the number of floating point operations.

Scondary storage for this algorithm is used to hold B which is overwritten with L, to hold A which is overwritten with C and later the band matrix, and to hold the matrices W and Y. Approximately $n^2/2$ storage is used for each of the four matrices, thus the total amount of secondary storage required is approximately $2n^2$. A Solid-State Storage Device (SSD) with 128 million words secondary storage with fast access on a Cray X-MP would allow problems of order up to 8000 before overflowing the SSD to slower secondary storage on disks.

The amount of I/O transfer between central memory and secondary storage storing the matrices A, C and the resulting band matrix is $((7/4)\,n^3)/p$ real words. The I/O transfer for the unit storing L, the Cholesky factor of B, is n^3/p. The total amount of I/O transfer is $((11/4)\,n^3)/p$.

The maximum central memory requirements for the new algorithm is

$$2np + \max(4p^2,\ np + 3n + p,\ np + n + p(p+1)),$$

where the three terms correspond to Cholesky factorization of B, reduction to banded form, and computation of the eigenvectors. A good working estimate for the amount of central memory required is $3np$. For the size of problems being considered (e.g., $n = 5000$ and $p = 50$) this is less than one million words of working storage.

The operation count for computing the eigenvalues of the eigenproblem in (1) by this approach is approximately $(11/3)\,n^3 + 10n^2p +$ lower order terms. This operation count comes from analyzing the 4 major components of the algorithm. The operation counts for these components are as follows:

Component	*Operation Count*
Cholesky factorization	$\frac{1}{3}n^3$
Reduction to standard form	$2n^3$
Block Householder transformations	$\frac{4}{3}n^3 + 2n^2p$
QR factorizations	$2n^2p$
Reduction of band matrix	$6n^2p$
Total	$\frac{11}{3}n^3 + 10n^2p$

For an execution with $n = 1492$ and $p = 50$ the monitored operation count was 13.3×10^9. The operation count computed by the above formula is 13.2×10^9 which is within 1% of the actual count.

The computational kernels used in this algorithm are Cholesky factorization, block solves with the Cholesky factor, QR factorization, matrix multiplication, and a banded eigenvalue solver. All of these kernels except for those used in solving the banded eigenproblem perform well on vector computers. In fact, they have com-

TABLE I

Comparison of Performance with In-core Methods (Execution Times in Seconds)

n	Number of eigenvectors	EISPACK	HSSXGV and HSSXG1	Block Lanczos
219	22	1.37	1.07	1.37
667	32	14.73	12.33	8.25
992	35	37.33	35.38	17.00
1496	35	108.54	88.74	47.70

putational rates in the 150 to 190 megaflop range on a Cray X-MP for the problem sizes being discussed in this application. The code is portable and can be implemented efficiently on other vector supercomputers, whenever high performance implementation of the Level 2 BLAS [2] are available.

3. Performance Results

A parameter study for the optimal choice of blocksize, p, was performed. A choice of 63 for the block size appears to be optimal on the Cray X-MP for the size of problems considered here. If a large number of eigenvectors will be computed a smaller blocksize might be in order.

Table I compared the combination of HSSXGV and HSSXG1 with the performance of the symmetric packed storage version of EISPACK and the block Lanczos algorithm described in [7] on the same eigenproblems in that paper. HSSXVG and HSSXG1 used block size 63 for all problems. The statement of the eigenproblem is to compute all eigenvalues and eigenvectors in the interval $[-1.0, 0.30]$.

HSSXGV and HSSXG1 not only allow larger problem sizes than the symmetric

TABLE II

Comparison of HSSXGV versus Block Lanczos on Matrix of Order 992 (Execution Times in Seconds)

Number of eigenvectors	HSSXGV and HSSXG1	Block Lanczos
20	33.77	21.20
40	35.70	20.62
60	37.78	27.02
80	39.69	30.78
100	41.32	39.35
120	43.66	53.91
140	45.58	62.35

TABLE III

Performance in MFLOPS

n	HSSXGV MFLOPS	HSSGX1 MFLOPS
219	74	51
667	127	66
992	128	72
1496	161	80

packed storage EISPACK path but have an 18% performance increase because of the block nature of the computations. Block Lanczos is indeed faster than HSSXGV and HSSXG1 for these eigenproblems. However, if more eigenvectors are required, HSSXGV and HSSXG1 is more efficient than block Lanczos. For the problem of order $n = 992$ the figures in Table II indicate that HSSXGV and HSSXG1 are faster than the block Lanczos code, if the number of required eigenvectors is increased to more than 100. We estimate that for the largest problem above, $n = 1496$, HSSXGV would become the algorithm of choice if 80 or more eigenvectors are required. Also HSSXGV computes all the eigenvalues and assumes no prior knowledge of the spectrum. Both Lanczos and the EISPACK path require an interval as input, and proceed to compute all the eigenvalues in the interval. Hence HSSXGV in Table III has computed additional information.

The overall computational rate for the problems in Table I is given in Table III. The highest performance with 161 MFLOPS was obtained on the largest problem. This number is quite remarkable since a significant amount of computation in the tridiagonal eigensolver is carried out in scalar mode. Assuming this computational rate the new software should be able to compute all the eigenvalues of an 8000 by 8000 dense generalized eigenvalue problem in about 3.3 hours on a Cray X-MP/24 with 128 Mword SSD.

4. Summary

Software for the out-of-core solution of the symmetric generalized eigenproblem has been developed and implemented. Because of its block nature, this software is more efficient on vector computers than a related in-core algorithm. If the number of required eigenpairs is large, in particular in applications, where all eigenvalues and vectors are required, the new software is more efficient than a previous code of the authors [7] based on the Lanczos algorithm. Most importantly, the new software allows efficient solution of problems too large to fit in central memory thus providing an important computational tool to the researches in quantum mechanics and other disciplines which generate large symmetric generalized eigenproblems.

A final word of caution: we have provided a convenient tool for finding all eigen-

values and eigenvectors of very large matrices. In many applications that we are aware of [4, 10, 11, 14] eigenvalues and vectors are only intermediate quantities in the computation. Sometimes the calculation of these intermediate quantities can be avoided altogether (see [11]). We encourage every potential user of our software to spend the extra time for analysis in order to evaluate whether the computation of the eigenvectors of a $10{,}000 \times 10{,}000$ matrix is really required.

REFERENCES

1. C. BISHOF AND C. VANLOAN, *SIAM J. Sci. Stat. Comput.* **8**, s2 (1987).
2. J. J. DONGARRA, J. DU CROZ, S. HAMMARLING, AND R. HANSON, "Extended Set of Fortran Basic Linear Algebra Subprograms," Argonne National Laboratory Report ANL-MSC-TM-41 (Revision 3), 1986 (unpublished).
3. J. J. DONGARRA AND D. C. SORENSON, *SIAM J. Sci. Stat. Comput.* **8**, s139 (1987).
4. A. J. FREEMAN, "All-electron density approach to the electronic structure at surfaces and interfaces," Report, Dept. of Physics, Northwestern University, Evanston, Illinois, 1986 (unpublished).
5. B. GARBOW, J. M. BOYLE, J. J. DONGARRA, AND C. MOLER, "Matrix Eigensystem Routines – *EISPACK Guide Extension*," Lecture Notes in Computer Sciences, Vol. 51 (Springer-Verlag, Berlin, 1977).
6. R. GRIMES, "Solving Systems of Large Dense Linear Equations," *J. Supercomput.* 1987 (to appear).
7. R. GRIMES, H. KRAKAUER, J. LEWIS, H. SIMON, AND S. WEI, *J. Comput. Phys.* **69**, 471 (1987).
8. R. GRIMES AND H. SIMON, "Solution of Large Dense Symmetric Generalized Eigenvalue Problems Using Secondary Storage," *ACM Trans. Math. Software*, (to appear).
9. R. GRIMES AND H. SIMON, "Subroutines for the Out-of-core Solution of Generalized Symmetric Eigenvalue Problem," Report ETA-TR-54, Boeing Computer Services, Seattle, 1987 (unpublished).
10. A. NAUTS AND R. E. WYATT, *Phys. Rev. Lett.* **51**, 2238 (1983).
11. A. NAUTS AND R. E. WYATT, *Phys. Rev. A* **30**, 872 (1984).
12. B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE, V. KLEMA, AND C. B. MOLER, "Matrix Eigensystem Routines – EISPACK Guide," Lecture Notes in Computer Sciences, Vol. 6 (Springer-Verlag, Berlin, 1976).
13. VectorPak Users Manual, Boeing Computer Services Document No. 20460-05010R1 (1987).
14. S. H. WEI AND H. KRAKAUER, *Phys. Rev. Lett.* **55**, 1200 (1985).

RECEIVED: May 22, 1987; REVISED: September 16, 1987

ROGER G. GRIMES
HORST D. SIMON†

Boeing Computer Services,
Engineering Technology Applications Division,
P. O. Box 24346, M/S 7L-21,
Seattle, Washington 98124-0346

† Present address: NAS Systems Division, NASA Ames Research Center, Moffett Field, CA 94035.